IN THE CLAIMS

Please amend the claims as follows:

1. (Original) A channel adapter comprising:

a host interface, the host interface operatively connected to a memory by a local bus, the memory containing at least one completion queue and at least one event queue;

a link interface, the link interface operatively connected to a network;

a packet processing engine, the packet processing engine moving data between the host interface and the link interface;

an address translation engine, the address translation engine translating a virtual address into a physical address of a translation protection table in the memory; and

a completion queue engine, the completion queue engine processing completion requests from the packet processing engine by writing the appropriate at least one of the at least one completion queue and at least one event queue,

wherein the packet processing engine is not impacted by any address translation functionality, completion queue accesses, or event queue accesses thereby significantly enhancing the performance of a channel adapter.

2. (Original) The adapter according to claim 1, wherein the network comprises one of an Infiniband network and a Next Generation Input Output (NGIO) network.

3. (Currently Amended) The adapter according to claim 1, wherein the host interface comprises at least one of a Peripheral Component Interconnect (PCI) ~~inteface~~ interface, PCI-X interface, and an [[a]] H-16 interface.

4. (Original) The adapter according to claim 1, wherein the packet processing engine converts the data from the host interface into packets to be sent to the link interface, and converts packets from the link interface into data to be sent to the host interface.

5. (Currently Amended) The adapter according to claim 1, wherein the completion queue engine comprises:

at least one local bus register, each at least one local bus register being programmable by at least one of an operating system and an application over the local bus;

at least one context memory, the at least one context memory storing context values for each at least one completion queue; and

a finite state machine, the finite state machine processing the completion requests from the packet processing engine and requests generated from the at least one local bus register.

6. (Original) The adapter according to claim 5, wherein the at least one local bus register comprises at least one of a completion queue base register, a completion queue entries register, a completion queue protection domain register, a completion queue interrupt enable register, a completion queue number register, and a completion queue control register, the contents of the completion queue base register, completion queue entries register, completion queue protection domain register, and completion queue interrupt enable register being stored in the context memory.

7. (Original) The adapter according to claim 6, wherein the completion queue engine further comprises at least one first working register, the at least one first working register being loaded with the contents of one of the completion queue base register, completion queue entries register, completion queue protection domain register, and completion queue interrupt enable register stored in the context memory while the finite state machine performs operations on the at least one context queue.

8. (Original) The adapter according to claim 7, wherein the contents loaded in the at least one first working register is stored back into the context memory after completing the operations.

9. (Original) The adapter according to claim 7, wherein the contents loaded in the at least one first working register is updated during the performing of operations, the updated contents being stored back into the context memory after completing the operations.

10. (Original) The adapter according to claim 6, wherein the completion queue control register contains an opcode, the opcode being used by at least one of the operating system and the application to one of enable and disable the at least one completion queue.

11. (Original) The adapter according to claim 10, wherein the at least one local bus register comprises at least one completion queue host entry register, the values in the at least one completion queue host entry register being added to the at least one completion queue indicated by the completion queue number register based on the opcode.

12. (Original) The adapter according to claim 5, wherein the at least one local bus register comprises at least one of an event queue base register, an event queue entries register, an event queue protection domain register, an event queue interrupt enable register, and an event queue control register.

13. (Original) The adapter according to claim 12, wherein the completion queue engine further comprises at least one second working register, the at least one second working register being loaded with the contents of one of the event queue base register, event queue entries register, event queue protection domain register, and event queue interrupt enable register while the finite state machine performs operations on the at least one event queue.

14. (Original) The adapter according to claim 13, wherein the event queue control register generates a request to update the values of the event queue base register, event queue entries register, and event queue protection domain register when the event queue control register is written to by at least one of the operating system and the application.

15. (Original) The adapter according to claim 5, wherein the at least one local bus register comprises at least one completion queue doorbell register, the at least one completion queue doorbell register allowing at least one of the operating system and the application to enable automatic event generation for at least one completion queue.

16. (Original) The adapter according to claim 1, wherein the address translation engine comprises an inbound request processor and a request completion processor, the inbound request processor receiving a request for address translation of the virtual address, the request completion processor sending a physical address of the memory associated with the virtual address and retrieved from the translation protection table to at least one of the packet processing engine and the completion queue engine in response to the request.

17. (Original) The adapter according to claim 16, wherein the inbound request processor comprises:

at least one switching device, the at least one switching device receiving the request for address translation from at least one of the packet processing engine, the completion queue engine, and the request completion processor;

at least one request register bank, each at least one request register bank comprising a request register, and a data register, the request register storing the request, the data register storing data received from at least one of the packet processing engine, the completion queue engine, and the request completion processor related to the request;

arbitration logic, the arbitration logic selecting between all outstanding requests and outputting a protection index of one of the requests, the arbitration logic associating a tag value with each request;

a local bus interface, the local bus interface receiving the protection index, checking whether the protection index is out of bounds, and adding the protection index with a base address of the translation protection table generating the physical address to the translation protection table; and

a request processor, the request processor sending errors related to the request to the request completion processor, the request processor sending a request and the physical address to the host interface to read the translation protection table.

18. (Original) The adapter according to claim 17, wherein the local bus interface comprises a size register and at least one base address register, the size register and at least one base address register being programmable over the local bus by at least one of the operating system and the application, the contents of the size register being compared to the protection index to determine whether the protection index is out of bounds, the base address register containing the base address of the translation protection table.

19. (Original) The adapter according to claim 17, wherein the data stored by the data register comprises one of a virtual address, a key, the protection index, and a protection domain.

20. (Original) The adapter according to claim 16, wherein the request completion processor comprises a decoder, at least one set of receive data buffers, arbitration logic, at least one staging register, permission checking logic, protection index calculation logic, and bounds checking logic.

21. (Original) The adapter according to claim 20, wherein each at least one set of receive data buffers comprises a read complete register and at least one data buffer, the read complete register signaling a request to the arbitration logic once all at least one data buffer for the set have been filled.

22. (Original) The adapter according to claim 20, wherein the decoder, at least one set of receive data buffers, and arbitration logic operate at a different clock speed than the at least one staging register, permission checking logic, protection index calculation logic, and bounds checking logic.

23. (Original) The adapter according to claim 19, further comprising a first valid register associated with the permission checking logic, a second valid register associated with the protection index calculation logic, and a third valid register associated with the bounds checking logic, each valid register containing an indication that the processing for the associated logic may begin.

24. (Original) A method for enhanced channel adapter performance comprising:

receiving a virtual address from a requester, the virtual address requiring translation to a physical address to a memory;

accessing a translation protection table using the virtual address to retrieve at least one data, one at least one data containing a first address;

checking the validity of the at least one data;

determining if a second access to the translation protection table is required based on the at least one data;

accessing the translation protection table using the first address to retrieve a second address if required; and

using one of the first address and a physical address to the memory contained at the second address to access the memory,

wherein the packet processing engine is not impacted by any address translation functionality, completion queue accesses, or event queue accesses thereby significantly enhancing the performance of a channel adapter.

25. (Original) The method according to claim 24, further comprising adding the virtual address to a base address of the translation protection table to get a physical address to the translation protection table.

26. (Original) The method according to claim 24, wherein the at least one data comprises at least one of window entries, region entries, and translation entries, the virtual address accessing at least one of window entries and region entries when part of a key request,

the virtual address accessing translation entries when part of a protection index request, each translation entry containing a physical address to the memory.

27. (Original) The method according to claim 26, wherein each window entry comprises a first entry containing a key, access rights, and protection information, a second entry containing the first address, and a third entry containing a third address.

28. (Original) The method according to claim 27, wherein the checking the validity comprises at least one of checking the access rights, comparing a protection key in the virtual address with the key in the first entry, and checking whether the virtual address is within the bounds of the translation protection table.

29. (Original) The method according to claim 27, further comprising adding the virtual address to the first address to generate the second address.

30. (Original) The method according to claim 26, wherein region entries comprise a first entry containing a key, access rights, a length of the number of associated translation entries following the first entry, and protection information, a second entry containing the first address, and a third entry containing a translation entry.

31. (Original) The method according to claim 30, wherein the checking the validity comprises at least one of checking the access rights, comparing a protection key in the virtual address with the key in the first entry, and checking whether the virtual address is within the bounds of the translation protection table.

32. (Original) The method according to claim 30, further comprising adding the virtual address to the first address to generate the second address.

33. (Original) A method for enhanced channel adapter performance comprising:

completing at least one work task by the channel adapter;

storing a completion status for each completed at least one work task in an entry in one at least one completion queue by a completion queue engine, each entry in each at least one completion queue being capable of storing multiple completion statuses;

receiving, at the completion queue engine, a completion request associated with at least one work task from a packet processing engine;

retrieving the completion status associated with the completion request from the one at least one completion queue by the completion queue engine; and

sending the completion status to the packet processing engine.

34. (Original) The method according to claim 33, further comprising storing notification of the storing of any completion status in one at least one completion queue in an entry in an event queue, the event queue containing an entry for each at least one completion queue.

35. (Original) The method according to claim 33, further comprising programming at least one register with at least one of a base address, a number of entries, a protection domain, an interrupt enable, and a queue number for one at least one completion queue.

36. (Original) The method according to claim 35, wherein the programming is performed by at least one of an operating system and an application.

37. (Original) The method according to claim 34, further comprising programming at least one register with at least one of a base address, a number of entries, a protection domain, and an interrupt enable for the event queue.

38. (Original) The method according to claim 37, wherein the programming is performed by at least one of an operating system and an application.

39. (Original) The method according to claim 33, further comprising programming at least one host register with data, the data being written into one at least one completion queue, the programming being performed by at least one of an operating system and an application.

40. (Original) The method according to claim 34, further comprising programming at least one doorbell register, the at least one doorbell register programmable with at least one of a memory mapped doorbell address and doorbell data, the doorbell data in each at least one doorbell register causing one associated at least one completion queue to generate an entry in the event queue when at least one completion status is stored in the one associated at least one completion queue.

41. (Original) The method according to claim 40, wherein the programming is performed by at least one of an operating system and an application by directly writing to the at least one doorbell register.

42. (Original) The method according to claim 40, wherein each doorbell address is a doorbell stride from each other, a doorbell stride being a relative offset of each doorbell address.

43. (Original) The method according to claim 40, wherein the doorbell stride is 4K address locations.

44. (Original) The method according to claim 40, wherein the doorbell stride is 16K address locations.

45. (Currently Amended) A computing device with enhanced channel adapter performance comprising:

a memory, the memory containing at least one translation protection table, at least one completion queue, at least one event queue, and at least one of a data buffer and a one work queue; and

a [[at]] channel adapter, the channel adapter comprising:

a packet processing engine, the packet processing engine moving data between a first interface and a second interface;

an address translation engine, the address translation engine translating a virtual address into a physical address of the at least one translation protection table in the memory; and

a completion queue engine, the completion queue engine processing completion requests from the packet processing engine by writing the appropriate at least one of the at least one completion queue and the at least one event queue, and

wherein the packet processing engine is not impacted by any address translation functionality, completion queue accesses, or event queue accesses thereby significantly enhancing the performance of the channel adapter.

46. (Original) The device according to claim 45, wherein the first interface comprises a host interface, the host interface operatively connected between the channel adapter and the memory by a local bus.

47. (Original) The device according to claim 46, wherein the completion queue engine comprises at least one local bus register, each at least one local bus register being programmable by at least one of an operating system and an application over the local bus.

48. (Original) The device according to claim 45, wherein the second interface comprises a link interface, the link interface operatively connected between the channel adapter and a network.

49. (Original) The device according to claim 48, wherein the network comprises one of an Infiniband network and a Next Generation Input Output (NGIO) network.